

brief contents

PART 1	INTRODUCTION TO NEO4J	1
	1 ■ A case for a Neo4j database	3
	2 ■ Data modeling in Neo4j	19
	3 ■ Starting development with Neo4j	32
	4 ■ The power of traversals	50
	5 ■ Indexing the data	64
PART 2	APPLICATION DEVELOPMENT WITH NEO4J	79
	6 ■ Cypher: Neo4j query language	81
	7 ■ Transactions	110
	8 ■ Traversals in depth	125
	9 ■ Spring Data Neo4j	147
PART 3	NEO4J IN PRODUCTION.....	181
	10 ■ Neo4j: embedded versus server mode	183
	11 ■ Neo4j in production	216

contents

<i>foreword</i>	<i>xi</i>
<i>preface</i>	<i>xiii</i>
<i>acknowledgments</i>	<i>xiv</i>
<i>about this book</i>	<i>xvi</i>
<i>about the authors</i>	<i>xx</i>
<i>about the cover illustration</i>	<i>xxii</i>

PART 1 INTRODUCTION TO NEO4J.....1

1 A case for a Neo4j database 3

1.1	Why Neo4j?	4
1.2	Graph data in a relational database	5
	<i>Querying graph data using MySQL</i>	6
1.3	Graph data in Neo4j	8
	<i>Traversing the graph</i>	9
1.4	SQL joins versus graph traversal on a large scale	11
1.5	Graphs around you	14
1.6	Neo4j in NoSQL space	14
	<i>Key-value stores</i>	15
	<i>Column-family stores</i>	15
	<i>Document-oriented databases</i>	16
	<i>Graph databases</i>	16
	<i>NoSQL categories compared</i>	16

- 1.7 Neo4j: the ACID-compliant database 17
- 1.8 Summary 18

2 *Data modeling in Neo4j* 19

- 2.1 What is a data model for Neo4j? 19
 - Modeling with diagrams: a simple example* 20
 - Modeling with diagrams: a complex example* 22
- 2.2 Domain modeling 24
 - Entities and properties* 24
- 2.3 Further examples 27
 - Underground stations example* 28 ▪ *Band members example* 29
- 2.4 Summary 31

3 *Starting development with Neo4j* 32

- 3.1 Modeling graph data structures 33
- 3.2 Using the Neo4j API 36
 - Creating nodes* 36 ▪ *Creating relationships* 39
 - Adding properties to nodes* 40 ▪ *Node type strategies* 42
 - Adding properties to relationships* 44
- 3.3 Node labels 46
- 3.4 Summary 48

4 *The power of traversals* 50

- 4.1 Traversing using the Neo4j Core Java API 51
 - Finding the starting node* 51 ▪ *Traversing direct relationships* 52 ▪ *Traversing second-level relationships* 54
 - Memory usage considerations* 57
- 4.2 Traversing using the Neo4j Traversal API 58
 - Using Neo4j's built-in traversal constructs* 58
 - Implementing a custom evaluator* 60
- 4.3 Summary 63

5 *Indexing the data* 64

- 5.1 Creating the index entry 65
- 5.2 Finding the user by their email 66
- 5.3 Dealing with more than one match 68
- 5.4 Dealing with changes to indexed data 69

- 5.5 Automatic indexing 70
 - Schema indexing* 70 ▪ *Auto-indexing* 73
- 5.6 The cost/benefit trade-off of indexing 75
 - Performance benefit of indexing when querying* 76
 - Performance overhead of indexing when updating and inserting* 77
 - Storing the index* 78
- 5.7 Summary 78

PART 2 APPLICATION DEVELOPMENT WITH NEO4J79

6 *Cypher: Neo4j query language* 81

- 6.1 Introduction to Cypher 82
 - Cypher primer* 82 ▪ *Executing Cypher queries* 84
- 6.2 Cypher syntax basics 89
 - Pattern matching* 89 ▪ *Finding the starting node* 93
 - Filtering data* 97 ▪ *Getting the results* 98
- 6.3 Updating your graph with Cypher 101
 - Creating new graph entities* 102 ▪ *Deleting data* 104
 - Updating node and relationship properties* 104
- 6.4 Advanced Cypher 105
 - Aggregation* 106 ▪ *Functions* 106 ▪ *Piping using the with clause* 108 ▪ *Cypher compatibility* 109
- 6.5 Summary 109

7 *Transactions* 110

- 7.1 Transaction basics 111
 - Adding in a transaction* 112 ▪ *Finishing what you start and not trying to do too much in one go* 113
- 7.2 Transactions in depth 114
 - Transaction semantics* 115 ▪ *Reading in a transaction and explicit read locks* 117 ▪ *Writing in a transaction and explicit write locks* 118 ▪ *The danger of deadlocks* 120
- 7.3 Integration with other transaction management systems 121
- 7.4 Transaction events 122
- 7.5 Summary 124

- ## 8 *Traversals in depth* 125
- 8.1 Traversal ordering 125
 - Depth-first* 126 ▪ *Breadth-first* 128 ▪ *Comparing depth-first and breadth-first ordering* 130
 - 8.2 Expanding relationships 132
 - StandardExpander* 132 ▪ *Ordering relationships for expansion* 135 ▪ *Custom expanders* 136
 - 8.3 Managing uniqueness 138
 - NODE_GLOBAL uniqueness* 138 ▪ *NODE_PATH uniqueness* 140 ▪ *Other uniqueness types* 142
 - 8.4 Bidirectional traversals 143
 - 8.5 Summary 145
- ## 9 *Spring Data Neo4j* 147
- 9.1 Where does SDN fit in? 148
 - What is Spring and how is SDN related to it?* 149
 - What is SDN good for (and not good for)?* 150
 - Where to get SDN* 150 ▪ *Where to get more information* 151
 - 9.2 Modeling with SDN 151
 - Initial POJO domain modeling* 151 ▪ *Annotating the domain model* 154 ▪ *Modeling node entities* 155
 - Modeling relationship entities* 158 ▪ *Modeling relationships between node entities* 160
 - 9.3 Accessing and persisting entities 163
 - Supporting Spring configuration* 163 ▪ *Neo4jTemplate class* 163 ▪ *Repositories* 165 ▪ *Other options* 168
 - 9.4 Object-graph mapping options 169
 - Simple mapping* 169 ▪ *Advanced mapping based on AspectJ* 173 ▪ *Object mapping summary* 175
 - 9.5 Performing queries and traversals 176
 - Annotated queries* 177 ▪ *Dynamically derived queries* 178
 - Traversals* 180
 - 9.6 Summary 180

PART 3 NEO4J IN PRODUCTION 181

10 *Neo4j: embedded versus server mode* 183

- 10.1 Usage modes overview 184
- 10.2 Embedded mode 186
 - Core Java integration* 186 ■ *Other JVM-based integration* 189
- 10.3 Server mode 190
 - Neo4j server overview* 191 ■ *Using the fine-grained Neo4j server REST API* 192 ■ *Using the Cypher Neo4j server REST API endpoint* 194 ■ *Using a remote client library to help access the Neo4j server* 196 ■ *Server plugins and unmanaged extensions* 197
- 10.4 Weighing the options 198
 - Architectural considerations* 199 ■ *Performance considerations* 201 ■ *Other considerations* 204
- 10.5 Getting the most out of the server mode 205
 - Avoid fine-grained operations* 206 ■ *Using Cypher* 207
 - Server plugins* 209 ■ *Unmanaged extensions* 212
 - Streaming REST API* 214
- 10.6 Summary 215

11 *Neo4j in production* 216

- 11.1 High-level Neo4j architecture 217
 - Setting the scene ...* 218 ■ *Disks* 218 ■ *Store files* 220
 - Neo4j caches* 222 ■ *Transaction logs and recoverability* 227
 - Programmatic APIs* 229
- 11.2 Neo4j High Availability (HA) 230
 - Neo4j clustering overview* 231 ■ *Setting up a Neo4j cluster* 234
 - Replication—reading and writing strategies* 237
 - Cache sharding* 241 ■ *HA summary* 243
- 11.3 Backups 244
 - Offline backups* 244 ■ *Online backups* 246
 - Restoring from backup* 248
- 11.4 Topics we couldn't cover but that you should be aware of 248
 - Security* 248 ■ *Monitoring* 249

11.5	Summary	249
11.6	Final thoughts	249
<i>appendix A</i>	<i>Installing Neo4j server</i>	<i>250</i>
<i>appendix B</i>	<i>Setting up and running the sample code</i>	<i>255</i>
<i>appendix C</i>	<i>Setting up your project to use SDN</i>	<i>261</i>
<i>appendix D</i>	<i>Getting more help</i>	<i>268</i>
	<i>index</i>	<i>269</i>